

Table 1 (cont')

```
/*  
 * strip path or prefix from pn, return len: pr_align()  
 */
```

```
static
```

```
5 stripname(pn) stripname
```

```
    char    *pn;    /* file name (may be path) */
```

```
{
```

```
    register char    *px, *py;
```

```
10
```

```
    py = 0;
```

```
    for (px = pn; *px; px++)
```

```
        if (*px == '/')
```

```
            py = px + 1;
```

```
15
```

```
    if (py)
```

```
        (void) strcpy(pn, py);
```

```
    return(strlen(pn));
```

```
}
```

```
20
```

```
25
```

```
30
```

```
35
```

```
40
```

```
45
```

```
50
```

```
55
```

```
60
```

Table 1 (cont')

```
/*
 * cleanup() -- cleanup any tmp file
 * getseq() -- read in seq, set dna, len, maxlen
 * g_calloc() -- calloc() with error checkin
5  * readjumps() -- get the good jumps, from tmp file if necessary
 * writejumps() -- write a filled array of jumps to a tmp file: nw()
 */
#include "nw.h"
10 #include <sys/file.h>

char    *jname = "/tmp/homgXXXXXX";      /* tmp file for jumps */
FILE    *fj;

15 int     cleanup();                      /* cleanup tmp file */
long    lseek();

/*
 * remove any tmp file if we blow
 */
20 cleanup(i)
    int    i;
{
    if (fj)
        (void) unlink(jname);
25     exit(i);
}

/*
 * read, return ptr to seq, set dna, len, maxlen
 * skip lines starting with ';', '<', or '>'
 * seq in upper or lower case
 */
30 char    *
getseq(file, len)
35     char    *file;    /* file name */
    int     *len;      /* seq len */
{
    char    line[1024], *pseq;
    register char    *px, *py;
    int     natgc, tlen;
    FILE    *fp;

    if ((fp = fopen(file, "r")) == 0) {
45         fprintf(stderr, "%s: can't read %s\n", prog, file);
        exit(1);
    }
    tlen = natgc = 0;
    while (fgets(line, 1024, fp)) {
50         if (*line == ';' || *line == '<' || *line == '>')
            continue;
        for (px = line; *px != '\n'; px++)
            if (isupper(*px) || islower(*px))
                tlen++;
    }
55     if ((pseq = malloc((unsigned)(tlen+6))) == 0) {
        fprintf(stderr, "%s: malloc() failed to get %d bytes for %s\n", prog, tlen+6, file);
        exit(1);
    }
    pseq[0] = pseq[1] = pseq[2] = pseq[3] = '\0';
60 }
```

Table 1 (cont')

...getseq

```
py = pseq + 4;
*len = tlen;
rewind(fp);

5 while (fgets(line, 1024, fp)) {
    if (*line == ';' || *line == '<' || *line == '>')
        continue;
    for (px = line; *px != '\n'; px++) {
10         if (isupper(*px))
            *py++ = *px;
        else if (islower(*px))
            *py++ = toupper(*px);
15         if (index("ATGCU",*(py-1)))
            natgc++;
    }
}
*py++ = '\0';
*py = '\0';
20 (void) fclose(fp);
dna = natgc > (tlen/3);
return(pseq+4);
}
```

```
25 char *
g_calloc(msg, nx, sz)
    char *msg;          /* program, calling routine */
    int nx, sz;          /* number and size of elements */
{
30     char *px, *calloc();

    if ((px = calloc((unsigned)nx, (unsigned)sz)) == 0) {
        if (*msg) {
35             fprintf(stderr, "%s: g_calloc() failed %s (n=%d, sz=%d)\n", prog, msg, nx, sz);
            exit(1);
        }
    }
    return(px);
}
```

g_calloc

```
40 /*
 * get final jmps from dx[] or tmp file, set pp[], reset dmax: main()
 */
```

```
readjmps()
45 {
    int fd = -1;
    int siz, i0, i1;
    register i, j, xx;

50     if (fj) {
        (void) fclose(fj);
        if ((fd = open(jname, O_RDONLY, 0)) < 0) {
            fprintf(stderr, "%s: can't open() %s\n", prog, jname);
            cleanup(1);
55         }
    }
    for (i = i0 = i1 = 0, dmax0 = dmax, xx = len0; i++) {
        while (1) {
60             for (j = dx[dmax].ijmp; j >= 0 && dx[dmax].jp.x[j] >= xx; j--);
        }
    }
}
```

readjmps